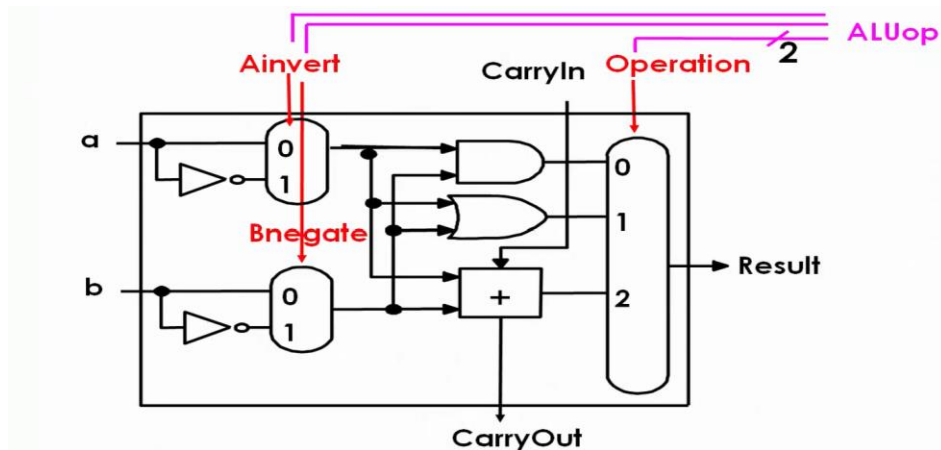


組別：_____ 簽名：_____

[group 9]

1. Derive the ALU op (4 bits) for the “NOR operation” and explain how it works on the following ALU:



Ans: 1100. The function “a nor b” is equal to “(not A) and (not B)”; therefore, the negations of A and B are both 1, and the final operation chooses the path 0, which is $(00)_2$. It is concluded that the ALUop of the nor operation is 1100.

[group 1]

2. In a Carry-Lookahead Adder (CLA), what is the conceptual difference between the **generate** (g_i) and **propagate** (p_i) signals, and what are their Boolean logic equations in terms of input bits a_i and b_i ?

Ans:

Generate ($g_i = a_i \cdot b_i$): Bit i creates a carry-out independently of whether there is an incoming carry c_i .

Propagate ($p_i = a_i + b_i$ or $a_i \oplus b_i$): Bit i passes an incoming carry c_i through to produce a carry-out (c_{i+1})

[group 2]

3. "In an N-bit ripple-carry adder, how does the ALU hardware know an overflow happened by comparing the carry signals at the sign bit (MSB)? Write the logic gate equation used for the Overflow output."

Ans:

"An overflow occurs if the carry going into the most significant bit (sign bit) doesn't match the carry coming out of it.

The hardware uses a XOR gate on these two carry bits: $\text{Overflow} = \text{CarryIn}[31] \text{ XOR } \text{CarryOut}[31]$

If both carries are 0 or both are 1, then XOR = 0 (No overflow)

If a carry is 1 and the other is 0, Then XOR = 1 (Overflow detected!)"

[group 5]

4. In an N-bit two's-complement ALU, overflow is detected by

$$\text{Overflow} = \text{CarryIn}[N-1] \text{ XOR } \text{CarryOut}[N-1]$$

where $\text{CarryIn}[N-1]$ is the carry into the MSB and $\text{CarryOut}[N-1]$ is the carry out of the MSB.

Let the MSB inputs of the adder be a and b, and the carry into the MSB be c.

Consider every possible combination of a and b.

Prove that the formula above is 1 exactly when overflow occurs. Use your proof to explain why overflow cannot occur when two operands with different signs are added.

Ans:

a = b = 0 (both positive): $\text{CarryOut} = 0$, and the sign bit of the sum = c. The result is wrongly negative (overflow) exactly when $c = 1$, which is exactly when $\text{CarryIn} \neq \text{CarryOut}$.

a = b = 1 (both negative): $\text{CarryOut} = 1$, and the sign bit of the sum = c. The result is wrongly positive (overflow) exactly when $c = 0$, which is again exactly when $\text{CarryIn} \neq \text{CarryOut}$.

a $\neq$ b (different signs): $a + b = 1$, so $\text{CarryOut} = c$ and the XOR is always 0. The result always lies between the two operands, so it is always representable and overflow cannot occur.

So the XOR is 1 if and only if overflow occurs.

[group 10]

5. Find the shortest sequence of MIPS instructions to determine there is a carry out from the addition of two registers, say, registers \$t3 and \$t4. Place a 0 or 1 in register \$t2 if the carry out is 0 or 1, respectively. (Hint: It can be done in two instructions.)

Ans:

```
addu $t2, $t3, $t4
sltu $t2, $t2, $t4
```

[group 13]

6. Consider the following 4-bit signed addition:

0111 + 0011

Try answering the following questions:

- (a) Perform the binary addition.
- (b) Represent the result in decimal.
- (c) Does signed overflow occur?
- (d) Explain *why* overflow does or does not occur using the signs of the two operands and the result.

Ans:

(a) 1010

(b) -6

(c) $7+3=10 > 7$, overflow occurred

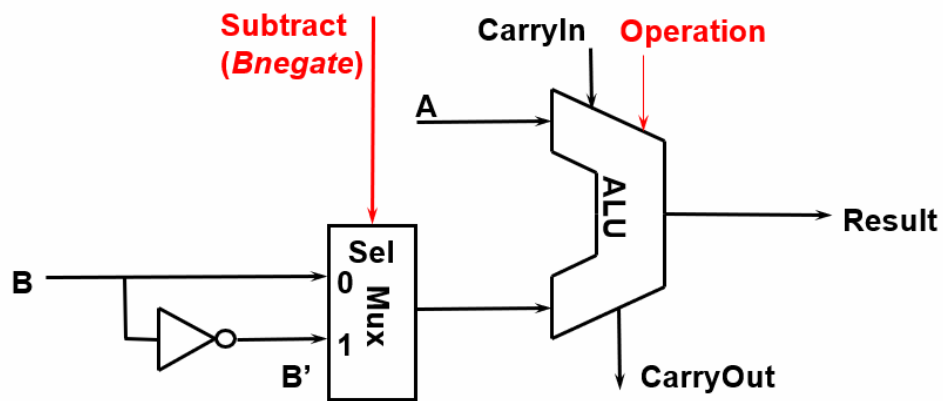
(d) We added two positive numbers together, and yet the binary addition yielded a negative result. For 4-bit signed integers, the valid range is between -8 to +7, and the operation produces $7+3=10$, which is outside the range, the sign flips and causes a signed overflow.

[group 7]

7. Suppose $A = 0110_2$ and $B = 0011_2$ are 4-bit binary numbers.

We want to use the ALU circuit shown above to compute $A-B$.

- (1) What is the bitwise complement B' of B ?
- (2) To perform subtraction, what should the value of B_{negate} be?
- (3) What is the value of $CarryIn$ of the least significant bit be?
- (4) How does ALU use “add” to generate a “sub” result? Please write down the process.
- (5) What are the values of the 4-bit Result and $CarryOut$?



Ans:

- (1) 1100
- (2) 1
- (3) 1
- (4) $A + B' + 1 = 0110 + 1100 + 0001$
- (5) Result = 0011₂, CarryOut = 1

[group 6]

8. Compare Ripple Carry Adder, Carry Lookahead Adder, and Carry Select Adder in terms of carry propagation, delay, and hardware cost.

Ans:

1. Ripple Carry Adder

- Carry Propagation: Each bit must wait for the carry from the previous bit.
- Delay: High delay due to sequential carry propagation. The worst-case delay is $2N$ gate delays.
- Hardware Cost: Low hardware cost and simple circuit design.

2. Carry Lookahead Adder

- Carry Propagation: Uses Generate (G) and Propagate (P) signals to calculate carry bits in advance.
- Delay: Faster than RCA because it reduces carry propagation delay.
- Hardware Cost: Higher hardware cost due to additional carry lookahead logic.

3. Carry Select Adder

- Carry Propagation: Computes two possible results in parallel, assuming $\text{CarryIn} = 0$ and $\text{CarryIn} = 1$.
- Delay: Reduces computation time by selecting the correct result using a MUX when the actual CarryIn arrives.
- Hardware Cost: Higher hardware cost because additional adders and multiplexers are required.

[group 4]

9. Consider a Carry Look-ahead Adder (CLA):
- (1) Write down the Boolean formulas for the Generate bit (g_i) and Propagate bit (p_i) in terms of operands A_i and B_i .
 - (2) Express carry inputs C_{in2} and C_{in3} in terms of g_i , p_i , and the initial carry-in C_{in0} .
 - (3) Why do commercial 32-bit adders typically use cascaded CLA or multi-level CLA instead of building a single 32-bit flat CLA?

Ans: (1) $g_i = A_i \cdot B_i$; $p_i = A_i \oplus B_i$

$$(2) C_{in2} = g_1 + (p_1 \cdot g_0) + (p_1 \cdot p_0 \cdot C_{in0})$$

$$C_{in3} = g_2 + (p_2 \cdot g_1) + (p_2 \cdot p_1 \cdot g_0) + (p_2 \cdot p_1 \cdot p_0 \cdot C_{in0})$$

(3) High gate fan-in causes heavy capacitive load, large delay, and routing difficulty; thus, 4-bit or 8-bit cascaded or multi-level designs balance speed and hardware cost.

[group 8]

10. Given the following inputs:

$$A = 1011$$

$$B = 0110$$

$$C_{in} = 1$$

For bit 0 only, find:

$$g_0, p_0, C_1$$

Use the formulas:

$$g_0 = A_0 \cdot B_0$$

$$p_0 = A_0 \text{ xor } B_0$$

$$C_1 = g_0 + p_0 C_{in}$$

Ans:

Look at the rightmost bit:

$$A_0 = 1, B_0 = 0, C_{in} = 1$$

Calculate generate:

$$g_0 = A_0 \cdot B_0, g_0 = 1 \cdot 0, g_0 = 0$$

Calculate propagate:

$$p_0 = A_0 \text{ xor } B_0, p_0 = 1 \text{ xor } 0, p_0 = 1$$

Calculate carry into bit 1:

$$C_1 = g_0 + p_0 C_{in}, C_1 = 0 + (1)(1), C_1 = 1$$

Final answer:

$$g_0 = 0, p_0 = 1, C_1 = 1$$

[group 3]

11. For an 8-bit adder, let $g_i = a_i \cdot b_i$, $p_i = a_i + b_i$, and $c_{i+1} = g_i + p_i \cdot c_i$. Define the group signals:

$$P_0 = p_3 \cdot p_2 \cdot p_1 \cdot p_0$$

$$G_0 = g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0$$

$$P_1 = p_7 \cdot p_6 \cdot p_5 \cdot p_4$$

$$G_1 = g_7 + p_7 \cdot g_6 + p_7 \cdot p_6 \cdot g_5 + p_7 \cdot p_6 \cdot p_5 \cdot g_4$$

1. Expand c_8 using only g , p and c_0 (one-level CLA).
2. Verify that the two-level form $c_8 = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot c_0$ equals your answer to (1).
3. If both are equivalent, why use two levels?

Ans:

- (1) Substituting the recurrence down to c_0 gives 9 terms:

$$\begin{aligned} c_8 = & g_7 + p_7 \cdot g_6 + p_7 \cdot p_6 \cdot g_5 + p_7 \cdot p_6 \cdot p_5 \cdot g_4 \\ & + p_7 \cdot p_6 \cdot p_5 \cdot p_4 \cdot g_3 + p_7 \cdot p_6 \cdot p_5 \cdot p_4 \cdot p_3 \cdot g_2 + p_7 \cdot p_6 \cdot p_5 \cdot p_4 \cdot p_3 \cdot p_2 \cdot g_1 + p_7 \cdot p_6 \cdot p_5 \cdot p_4 \cdot p_3 \cdot p_2 \cdot p_1 \cdot g_0 \\ & + p_7 \cdot p_6 \cdot p_5 \cdot p_4 \cdot p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot c_0 \end{aligned}$$

- (2) Distribute P_1 over G_0 , and regroup $P_1 \cdot P_0$:

$$P_1 \cdot G_0 = p_7 \cdot p_6 \cdot p_5 \cdot p_4 \cdot g_3 + p_7 \cdot p_6 \cdot p_5 \cdot p_4 \cdot p_3 \cdot g_2 + p_7 \cdot p_6 \cdot p_5 \cdot p_4 \cdot p_3 \cdot p_2 \cdot g_1 + p_7 \cdot p_6 \cdot p_5 \cdot p_4 \cdot p_3 \cdot p_2 \cdot p_1 \cdot g_0$$

$$P_1 \cdot P_0 \cdot c_0 = p_7 \cdot p_6 \cdot p_5 \cdot p_4 \cdot p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot c_0$$

Adding G_1 (4 terms) gives the same 9 terms as (1). Only the distributive law and the associativity of AND are used, so the two forms are equivalent. ■

- (3) The one-level c_{16} needs 17-input gates. The two-level version needs at most 5-input gates, and each P and G is computed once and shared by several carries.